# Digital Design
# Finite State Machine Implementation

Student Name: Juliet Chinyere Nkwor

Student ID: 2340240

Tutor: Steven Quigley

April 30, 2022

## Introduction

The primary objective of this assignment is to implement a sophisticated digital circuit on a programmable logic device. For this assignment, we are required to implement a finite state machine design using the Vivado design package. Modifications to the basic synchronous design will be required to allow the machine to be forced into the not-allowed states for testing purposes.

A synchronous finite state machine is to be designed and implemented. Both the transition function logic and output function logic will be designed.

The machine will give one output pulse equal in length to a clock cycle, after a pre-defined number of clock transitions. The finite state machine should be designed to count cyclically through a pre-defined series of states corresponding to standard binary notation whilst skipping 'not allowed' states (e.g. for a non-existent specification: S15, S8, S9, S5, S3, S1, S10, S14, S2, S11, etc. = {1111, 1000, 1001, 0101, 0011, 0001, 1010} etc.).

The second part is that the finite state machine is to re-synchronise within one clock pulse to an allowed state should the system enter a 'not allowed' state at switch-on time. The state into which the finite state machine re-synchronises should be chosen to minimise the complexity of the circuitry.

The design of the finite state machine is targeted at using a Nexys 4 board programmed in VHDL.

## Implementation Process

The given sequence is **14,2,13,7,1,12,10,9,15,5,3,11**. The number of states is not an exact power of two which leads us to 4 unused **states 0, 4, 6, 8**. We will discuss more on the unused state in part B. The output is active at 7.

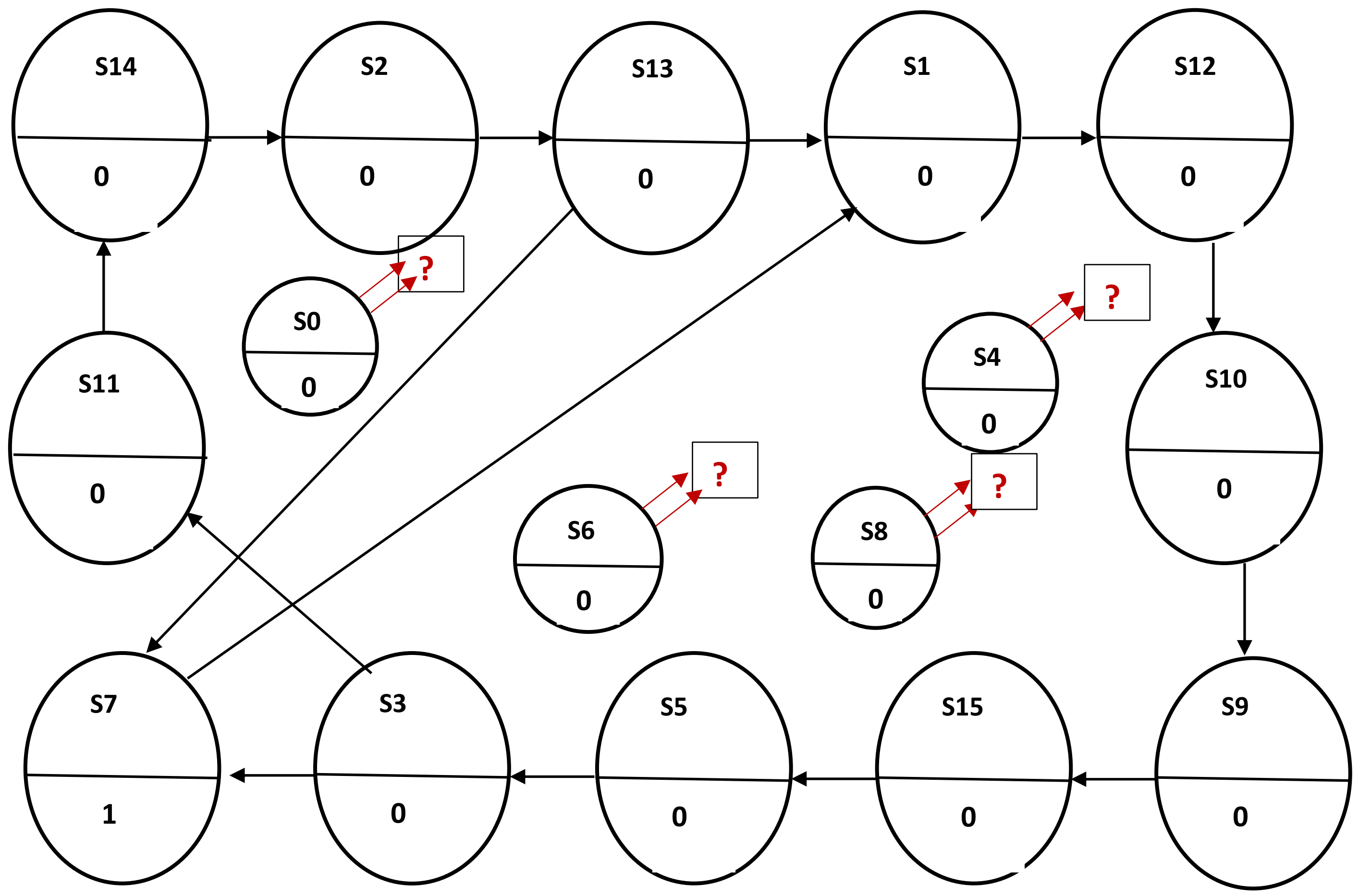
To demonstrate our work on the NEXYs 4 device, two buttons are used. The BTND is for reset and BTNU is for set. The LEDs at H17, K15, J13, and N14 are used to indicate the state you are in. The LED at U16 is the clock and the LED at V17 is the active indicator which in my case is 7.
When you click on both buttons, the system is expected to transition through the given sequence at every rising clock edge.

### Method

1. The machine starts at state S14, the LEDs at N14, J13 and K15 indicates that we are at S14.
2. Click on BTND, the BTNU, the LED at K15 comes on to indicate that we have transitioned to state S2.
3. Click on BTND, the BTNU, the LED at N14, J13 and H17 comes to indicate that we have transitioned to state S13.
4. Click on BTND, the BTNU, the LED at J13, K15 and H17 comes up to indicate that we have transitioned to state S7. The V17 will also come on to indicate that we are in the active state.
5. Click on BTND, the BTNU, the LED at H17 comes on to indicate that we have transitioned to state S1.
6. Click on BTND, the BTNU, the LED at N14 and J13 comes on to indicate that we have transitioned to state S12.
7. Click on BTND, the BTNU, the LED at N14 and K15 comes on to indicate that we have transitioned to state S10.
8. Click on BTND, the BTNU, the LED at N14, H17 comes on to indicate that we have transitioned to state S9.
9. Click on BTND, the BTNU, the LED at N14, J13, K15 and H17 comes on to indicate that we have transitioned to state S15.
10. Click on BTND, the BTNU, the LED at J13, and H17 comes on to indicate that we have transitioned to state S5.
11. Click on BTND, the BTNU, the LED at K15 and H17 comes on to indicate that we have transitioned to state S3.
12. Click on BTND, the BTNU, the LED at N14, K15 and H17 comes on to indicate that we have transitioned to state S11. From S11, you go back to S14.

## State Diagram

The machine transits cyclically through the sequence given sequence: **14,2,13,7,1,12,10,9,15,5,3,11**



## State Table

The tabular translation of the transition of the sequence at the rising cl;ock.

| **Present State** | **Next State** | |
|---|---|---|
| | **Clock = 0** | **Clock = 1** |
| **S1** | S1 | S12 |
| **S2** | S2 | S13 |
| **S3** | S3 | S11 |
| **S5** | S5 | S3 |
| **S7** | S7 | S1 |
| **S9** | S9 | S15 |
| **S10** | S10 | S9 |
| **S11** | S11 | S14 |
| **S12** | S12 | S10 |
| **S13** | S13 | S7 |
| **S14** | S14 | S2 |
| **S15** | S15 | S5 |

## Code Assignment

There are 12 states in our state diagram, and we need 4 state variables. we have $2^4 = 16$ states once the machine is implemented. So now there are 4 states that must be routed somewhere suitable. The specification implies that we must route these to minimize circuit complexity.

We will use 4-bit code assignment, and labels Qd, Qc, Qb, Qa as tabulated below.

| **State** | **Qd** | **Qc** | **Qb** | **Qa** |
|---|---|---|---|---|
| **S0** | 0 | 0 | 0 | 0 |
| **S1** | 0 | 0 | 0 | 1 |
| **S2** | 0 | 0 | 1 | 0 |
| **S3** | 0 | 0 | 1 | 1 |
| **S4** | 0 | 1 | 0 | 0 |
| **S5** | 0 | 1 | 0 | 1 |
| **S6** | 0 | 1 | 1 | 0 |
| **S7** | 0 | 1 | 1 | 1 |
| **S8** | 1 | 0 | 0 | 0 |
| **S9** | 1 | 0 | 0 | 1 |
| **S10** | 1 | 0 | 1 | 0 |
| **S11** | 1 | 0 | 1 | 1 |
| **S12** | 1 | 1 | 0 | 0 |
| **S13** | 1 | 1 | 0 | 1 |
| **S14** | 1 | 1 | 1 | 0 |
| **S15** | 1 | 1 | 1 | 1 |

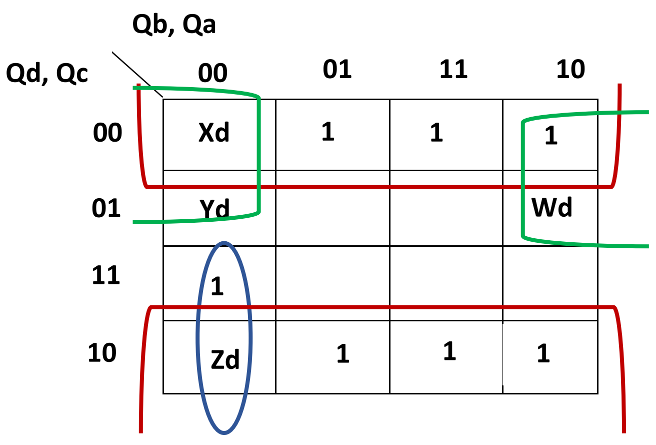
## Transition Table

For this, we will calculate/work out the D inputs that must be ready for the clock edge needed to give us the required values of **Qnew**. We used symbolic names X, Y, W, Z with red colour to represent the not-allowed states.

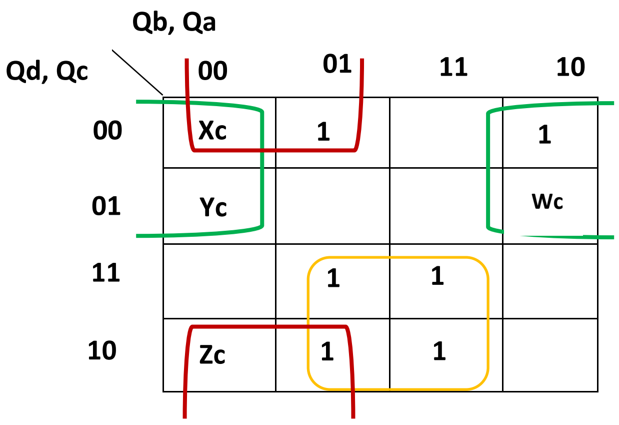
| **Present State** | | | | | **Next State** | | | | | **D values required for next state** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **State** | **Qd** | **Qc** | **Qb** | **Qa** | **State** | **Qd new** | **Qc new** | **Qb new** | **Qa new** | **Dd** | **Dc** | **Db** | **Da** |
| **S0** | 0 | 0 | 0 | 0 | **?** | **Xd** | **Xc** | **Xb** | **Xa** | **Xd** | **Xc** | **Xb** | **Xa** |
| **S1** | 0 | 0 | 0 | 1 | **S12** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **S2** | 0 | 0 | 1 | 0 | **S13** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **S3** | 0 | 0 | 1 | 1 | **S11** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| **S4** | 0 | 1 | 0 | 0 | **?** | **Yd** | **Yc** | **Yb** | **Ya** | **Yd** | **Yc** | **Yb** | **Ya** |
| **S5** | 0 | 1 | 0 | 1 | **S3** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| **S6** | 0 | 1 | 1 | 0 | **?** | **Wd** | **Wc** | **Wb** | **Wa** | **Wd** | **Wc** | **Wb** | **Wa** |
| **S7** | 0 | 1 | 1 | 1 | **S1** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **S8** | 1 | 0 | 0 | 0 | **?** | **Zd** | **Zc** | **Zb** | **Za** | **Zd** | **Zc** | **Zb** | **Za** |
| **S9** | 1 | 0 | 0 | 1 | **S15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **S10** | 1 | 0 | 1 | 0 | **S9** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| **S11** | 1 | 0 | 1 | 1 | **S14** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| **S12** | 1 | 1 | 0 | 0 | **S10** | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **S13** | 1 | 1 | 0 | 1 | **S7** | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| **S14** | 1 | 1 | 1 | 0 | **S2** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **S15** | 1 | 1 | 1 | 1 | **S5** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

## Extraction of Implicants

We will use Karnaugh Maps to extract the Implicants.

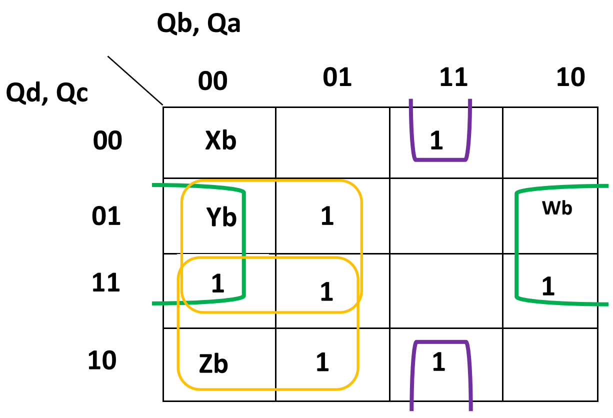


$$Dd = \overline{Qc} + \overline{Qd} . \overline{Qa} + \overline{Qb}. \overline{Qa} . Qd$$



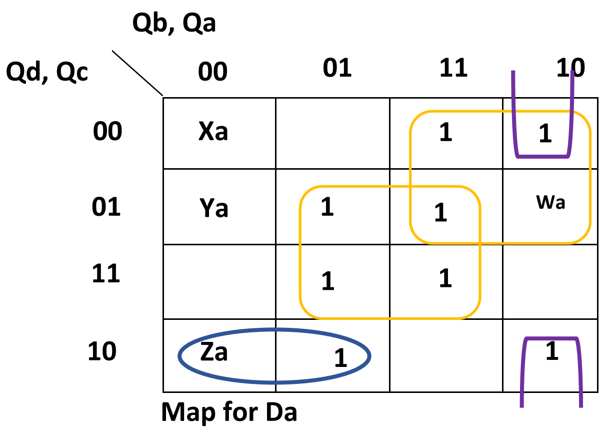
| Qd, Qc \ Qb, Qa | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | Xc | 1 |  | 1 |
| 01 | Yc |  |  | Wc |
| 11 |  | 1 | 1 |  |
| 10 | Zc | 1 | 1 |  |

Dc = $\overline{Qd}.\ \overline{Qa} + \overline{Qc}\ .\ \overline{Qb} + Qd.\ Qa$



| Qd, Qc \ Qb, Qa | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | Xb |  | 1 |  |
| 01 | Yb | 1 |  | Wb |
| 11 | 1 | 1 |  | 1 |
| 10 | Zb | 1 | 1 |  |

Db = $Qc.\ \overline{Qa} + Qc.\ \overline{Qb} + Qd.\ \overline{Qb} + \overline{Qc}.\ Qb.Qa$



| Qd, Qc \ Qb, Qa | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | Xa |  | 1 | 1 |
| 01 | Ya | 1 | 1 | Wa |
| 11 |  | 1 | 1 |  |
| 10 | Za | 1 |  | 1 |

Map for Da

Da $= \overline{Qd}\ .\ Qb + Qc.\ Qa + Qd.\ \overline{Qc}.\ \overline{Qb} + \overline{Qc}.\ Qb.\ \overline{Qa}$

## PART B

The deliverables of this assignment require that you can demonstrate the counter recovering from a not-allowed state. For testing purposes, the 'JAM_ENABLE' signal will be activated before a clock pulse. This will allow the data placed on the input lines JAM_A to JAM_D to be clocked onto the output pins. The clock pulses should be derived from a suitable switch de-bounce.

### Method

The JAM_ENABLE switch is switch 5 (R17).

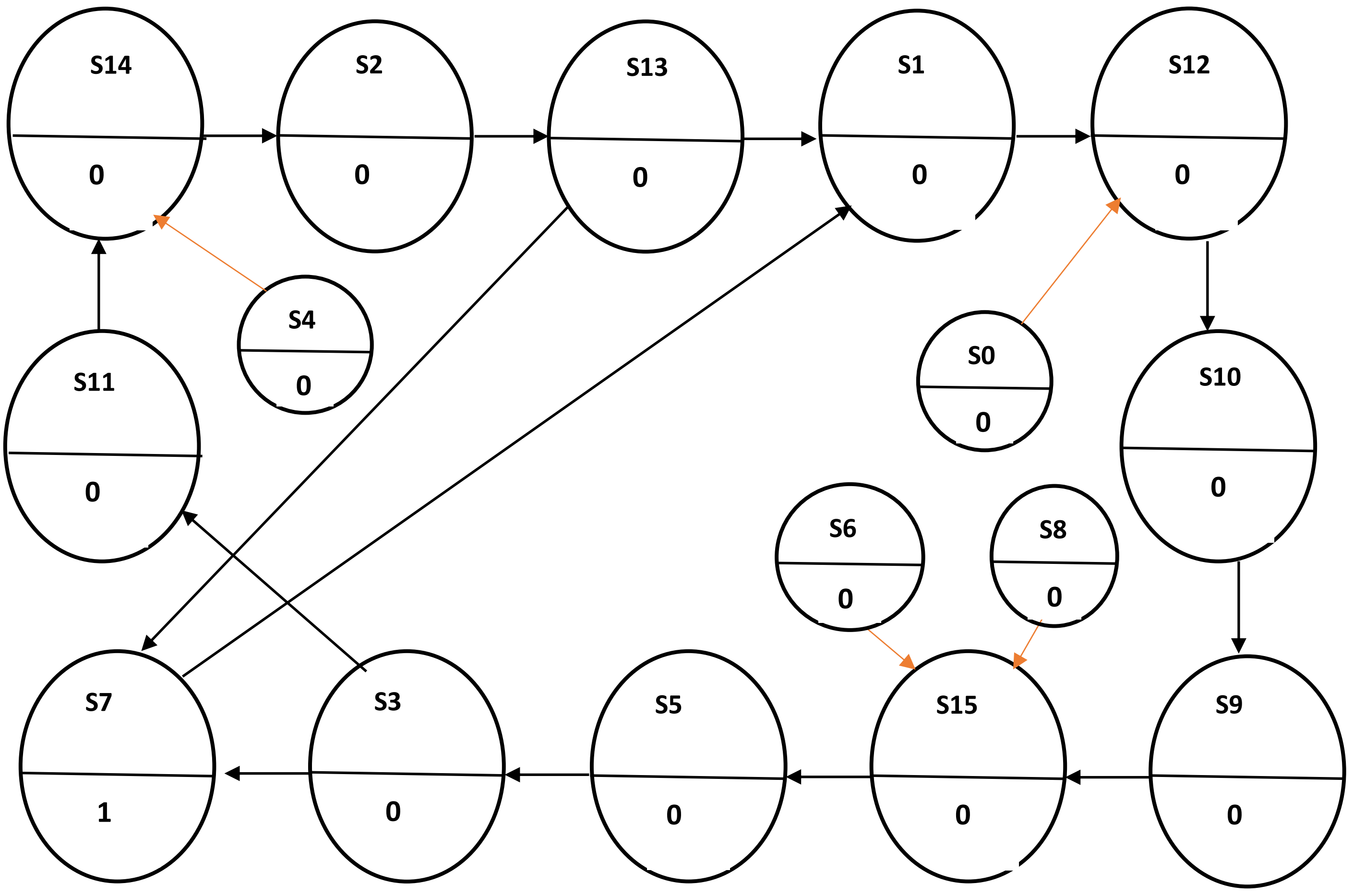
1. Set the JAM_ENABLE switch.
2. Set the J15, L16, 13 and R15 switches to read zero.
3. Click on the BTND and BTNU button.
4. De-select the JAM_ENABLE switch
5. Click on the BTND and BTNU button again.
6. The LEDs at R18 and J13 indicates that the unallowed state S0 and been forced to S12.
7. The process is repeated for S4, S6, S8 by setting switches to the appropriate unallowed state number.

The D-inputs of the unallocated state destinations are:

**X = 1 1 0 0, Y = 1 1 1 0, W = 1 1 1 1, Z = 1 1 1 1**

## Revised State Diagram

We have now mapped out our new transitions, ensuring that an unallocated state does not transition to another unallocated state.



## Revised Transition Table

The transition table below has been updated to reflect the states the not-allowed state will transit to.

| **Present State** | | | | | **Next State** | | | | | **D values required for next state** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **State** | **Qd** | **Qc** | **Qb** | **Qa** | **State** | **Qd new** | **Qc new** | **Qb new** | **Qa new** | **Dd** | **Dc** | **Db** | **Da** |
| **S0** | 0 | 0 | 0 | 0 | **S12** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **S1** | 0 | 0 | 0 | 1 | **S12** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **S2** | 0 | 0 | 1 | 0 | **S13** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **S3** | 0 | 0 | 1 | 1 | **S11** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| **S4** | 0 | 1 | 0 | 0 | **S14** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| **S5** | 0 | 1 | 0 | 1 | **S3** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| **S6** | 0 | 1 | 1 | 0 | **S15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **S7** | 0 | 1 | 1 | 1 | **S1** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **S8** | 1 | 0 | 0 | 0 | **S15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **S9** | 1 | 0 | 0 | 1 | **S15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **S10** | 1 | 0 | 1 | 0 | **S9** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| **S11** | 1 | 0 | 1 | 1 | **S14** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| **S12** | 1 | 1 | 0 | 0 | **S10** | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **S13** | 1 | 1 | 0 | 1 | **S7** | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| **S14** | 1 | 1 | 1 | 0 | **S2** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **S15** | 1 | 1 | 1 | 1 | **S5** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

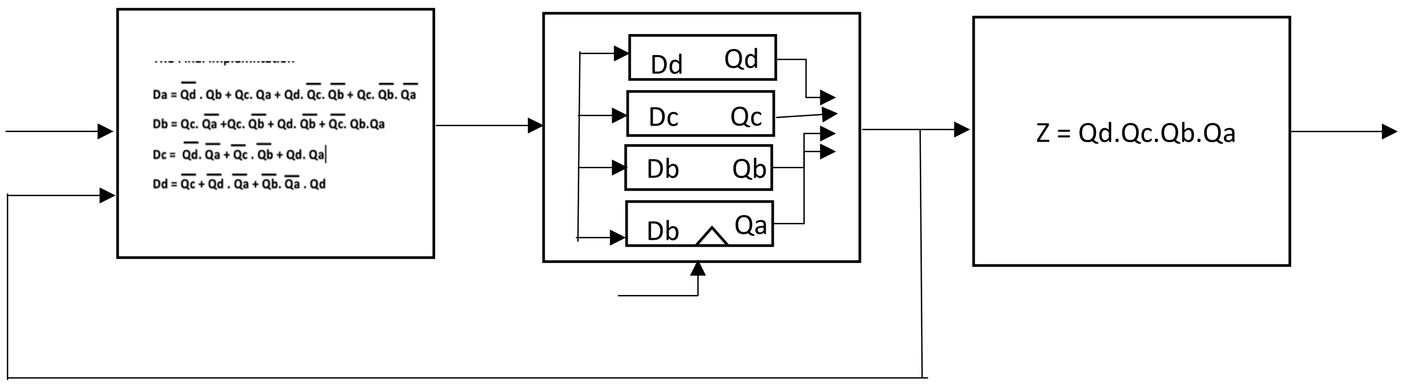
## The Final Implemntation

$$Da = \overline{Qd} \,.\, Qb + Qc.\, Qa + Qd.\, \overline{Qc}.\, \overline{Qb} + Qc.\, \overline{Qb}.\, \overline{Qa}$$

$$Db = Qc.\, \overline{Qa} + Qc.\, \overline{Qb} + Qd.\, \overline{Qb} + \overline{Qc}.\, Qb.Qa$$

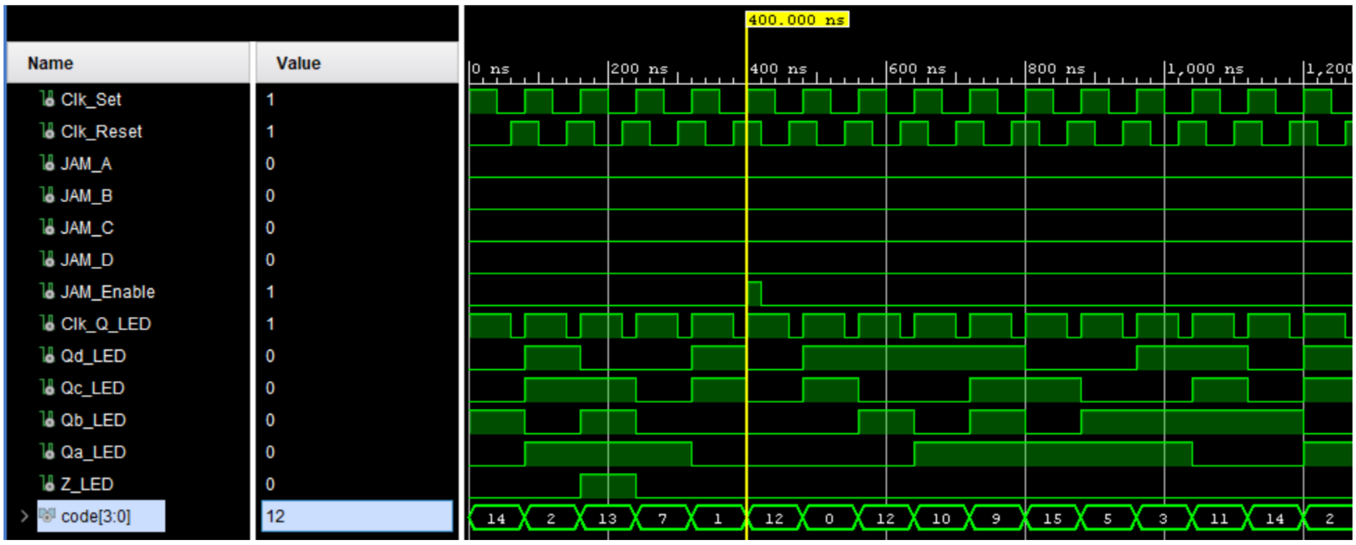
$$Dc = \overline{Qd}.\, \overline{Qa} + \overline{Qc} \,.\, \overline{Qb} + Qd.\, Qa$$

$$Dd = \overline{Qc} + \overline{Qd} \,.\, \overline{Qa} + \overline{Qb}.\, \overline{Qa} \,.\, Qd$$


Da = Qd . Qb + Qc. Qa + Qd. Qc. Qb + Qc. Qb. Qa
Db = Qc. Qa +Qc. Qb + Qd. Qb + Qc. Qb.Qa
Dc = Qd. Qa + Qc . Qb + Qd. Qa
Dd = Qc + Qd . Qa + Qb. Qa . Qd
Dd
Qd
Dc
Qc
Db
Qb
Db
Qa
Z = Qd.Qc.Qb.Qa

## Simulation

The result shows that the machine transits through the sequence at every rising clock edge as expected. Also, when the JAM_ENABLED is clocked, the machine transits to the specified recovery state.



| Name | Value |
|---|---|
| Clk_Set | 1 |
| Clk_Reset | 1 |
| JAM_A | 0 |
| JAM_B | 0 |
| JAM_C | 0 |
| JAM_D | 0 |
| JAM_Enable | 1 |
| Clk_Q_LED | 1 |
| Qd_LED | 0 |
| Qc_LED | 0 |
| Qb_LED | 0 |
| Qa_LED | 0 |
| Z_LED | 0 |
| code[3:0] | 12 |

## Conclusion

In this work we were able to demonstrate what to do when the number of states is not a $2^n$ number and how to optimise the destination of unallocated states to minimise the logic complexity.

## References

Diligent (2004). *Revised edition? Nexys4 DDR FPGA Board Reference Manual* [DX Reader version]. Retrieved from https://digilent.com/